

OpenMP and the Stommel Model

NPACI Parallel Computing Seminars
San Diego Supercomputing Center

Talk Overview

- What we look for when using OpenMP
- The Stommel model is a good candidate for OpenMP
- Timings for serial version
- Adding our directive
- What happened?
- Fixing the problem
- Final runtimes

What we look for when using OpenMP

- Do a profile to find time dominate routines
- Better yet find time dominate loop
 - Best case:
 - No dependency between iterations
 - A lot of work in each iteration
 - May want to rewrite your loops
 - Remove dependencies
 - Merge loops to reduce synchronization costs
- Can you rewrite your program in a task level parallel fashion?

Why OpenMP?

- Stommel Model well suited for OpenMP
 - Single nested do loop dominates the runtime

```
subroutine do_jacobi(psi,new_psi,diff,i1,i2,j1,j2)
...
...
do j=j1,j2
  do i=i1,i2
    new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
                  a3*psi(i,j+1) + a4*psi(i,j-1) - &
                  a5*for(i,j)
    diff=diff+abs(new_psi(i,j)-psi(i,j))
  enddo
enddo
psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
```

- No dependencies between iterations

Serial runtime

- Run on Dec Alpha heinlein
 - Fast processors 533 MHz Alpha
 - 4 SM processors / node
 - Good OpenMP support
 - f90 stf_00.f90 -O4 -o str_00.exe
 - Took out write_grid routine
- "Standard" Input:
200 200
2000000 2000000
1.0e-9 2.25e-11 3.0e-6
75000

Dec Alpha Stommel Model Serial Time

```
f90 stf_00.f90 -O4 -o str_00.exe
timex str_00.exe < st.in
run time = 194.52
real 194.6
user 194.4
sys 0.0
```

A little better than the IBM SP

run time = 265.41

Our directives

- Put directive around outer loop
 - A1-A5 are private and defined outside the loop
 - Diff is a reduction variable
 - We want i to be private also
- Our directive is:

```
$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
                      firstprivate(a1,a2,a3,a4,a5) reduction(:diff)
do j=j1,j2
  do i=i1,i2
    new_psi(i,j)= a1*psi(i+1,j) + a2*psi(i-1,j) + &
                  a3*psi(i,j+1) + a4*psi(i,j-1) - &
                  a5*for(i,j)
    diff=diff+abs(new_psi(i,j)-psi(i,j))
  enddo
enddo
```

Dec Alpha Stommel Model OpenMP Time 4 threads

```
f90 -omp stf_00.f90 -O4 -o stf_00.omp
timex str_00.exe < st.in
run time = 363.95
real 364.2
user 1451.6
sys 0.5
```

Serial runtime:

```
run time = 194.52
```

Ouch! What happened?

Ouch! What Happened?

```
!$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
    firstprivate(a1,a2,a3,a4,a5) reduction(:diff)
    do j=j1,j2
        do i=i1,i2
            new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
                            a3*psi(i,j+1) + a4*psi(i,j-1) - &
                            a5*for(i,j)
            diff=diff+abs(new_psi(i,j)-psi(i,j))
        enddo
    enddo
!$OMP END PARALLEL DO
    psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
```

- Program spent much time in synchronization after do loops at line:
 - `psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)`
 - This should not happen but shows immaturity of the compiler even though DEC compiler is one of the bests
 - OpenMP does not yet handle array syntax

Work around

- Rewrite the array syntax line and insert directives

```
!     psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
 !$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
   do j=j1,j2
     do i=i1,i2
       psi(i,j)=new_psi(i,j)
     enddo
   enddo
 !!!$OMP END PARALLEL DO
```

Dec Alpha Stommel Model OpenMP Time 4 threads modified source

```
f90 -omp stf_00.f90 -O4 -o stf_00.omp
timex str_00.exe < st.in
run time = 50.10
real 50.3
user 200.3
sys 0.1
```

Serial runtime:

```
run time = 194.52
```

Factor of 4 Speedup!

Our final subroutine header stuff

```
subroutine do_jacobi(psi,new_psi,diff,i1,i2,j1,j2)
    use numz
    use constants
    implicit none
    integer,intent(in) :: i1,i2,j1,j2
    real(b8),dimension(i1-1:i2+1,j1-1:j2+1):: psi
    real(b8),dimension(i1-1:i2+1,j1-1:j2+1):: new_psi
    real(b8) diff
    integer i,j
    real(b8) y
    diff=0.0_b8
```

Important part

```
!$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
    firstprivate(a1,a2,a3,a4,a5) reduction(:diff)
do j=j1,j2
    do i=i1,i2
        new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
                      a3*psi(i,j+1) + a4*psi(i,j-1) - &
                      a5*for(i,j)
        diff=diff+abs(new_psi(i,j)-psi(i,j))
    enddo
enddo
 !$OMP END PARALLEL DO
!     psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
 !$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
    do j=j1,j2
        do i=i1,i2
            psi(i,j)=new_psi(i,j)
        enddo
    enddo
 !$OMP END PARALLEL DO
end subroutine do_jacobi
```

We can combine MPI and OpenMP

- The most likely programming methodology for our new teraflop SP machine
- Works well on the our DEC cluster
 - 2 nodes vinge.sdsc.edu and helein.sdsc.edu
 - Run two MPI processes
 - Each MPI process runs 4 MPI threads
 - We simply combine our Stommel model version stf_002.f with the OpenMP enabled subroutine

Dec Alpha running OpenMP & MPI

```
f90 -O4 -free -omp -lmpi -lfmpi omp_02.F \
-o omp_02
```

We have MPI
and OpenMP

```
cat hosts
```

```
vinge.sdsc.edu
```

```
heinlein.sdsc.edu
```

Host file required to
run across 2 nodes

```
cat do_it
```

```
#!/bin/csh -f
setenv OMP_NUM_THREADS 4
omp_02 < st.in
```

Script file to set
threads and run job

```
dmpirun -np 2 -hf hosts do_it
```

Our run command

run time = 44.12

New IBM SP running OpenMP & MPI

```
mpxlf_r -qfree=f90 -O3 -qtune=pwr3 -qarch=pwr3 -qsmp  
omp_02.f -o omp_02
```

```
cat run_both  
#! /usr/bin/ksh -f  
printenv | grep MP_  
export OMP_NUM_THREADS=4  
omp_02 < st.in
```

We have MPI
and OpenMP

Script file to set 4
threads and run job

This script is run
from LoadLeveler

New IBM SP running OpenMP & MPI

```
$ cat bonk2
#!/usr/bin/ksh
#@ environment = COPY_ALL; MP_EUILIB = us
#@initialdir = /nhome/tkaiser
#@ output = $(jobid).out
#@ error = $(jobid).out
#@ job_type = parallel
#@ class = high
#@ network.MPI = css0,not_shared,US
#@ node_usage = not_shared
#@ node = 2
#@ tasks_per_node = 1
#@ wall_clock_limit = 1:00:00
#@ notify_user = tkaiser@sdsc.edu
#@ notification = error
#@ notification = complete
#@ queue
poe run_both
```

← Don't share the nodes

```
cat run_both
#!/usr/bin/ksh -f
printenv | grep MP_
export OMP_NUM_THREADS=4
omp_02 < st.in
```

← Run on 2 nodes with 1 MPI task/node

← Script file to set threads and run job

Summary

- Simple changes enabled a factor of 4 speed up
- Need to be careful because of immaturity of the compilers
- We can combine OpenMP with MPI to take advantage of hybrid machines